Arrays

Lets look at ways to store, and manipulate, multiple values of the same type.

The most common way to do this, in Java, is with an array.

Arrays

An array is a data structure, that allows you to store a sequence of values, all of the same type.

You can have arrays for any primitive type, like ints, doubles, booleans, or any of the 8 primitives we've learned about.

You can also have arrays for any class.

Arrays

Elements in an array are indexed, starting at 0.

If we have an array, storing five names, conceptually it looks as shown here.

Index	**0**	**1**	**2**	**3**	**4**
Stored values in an array with 5 elements	"Andy"	"Bob"	"Charlie"	"David"	"Eve"

The first element is at index 0, and is Andy.

The last element in this array is at index 4, and has the String value Eve.

Declaring an Array

When you declare an array, you first specify the type of the elements you want in the array.

Then you include square brackets in the declaration, which is the key for Java, to identify the variable as an array.

The square brackets can follow the type as shown in the first two examples.

This is much more common.

The brackets can also be after the variable name, as shown in the last example.

You don't specify a size, in the array declaration itself.

Array Variable Declaration
`int[] integerArray;`
`String[] nameList;`
`String courseList[];`

Instantiating an Array

Array Creation	Object Creation
`int[] integerArray = new int[10];`	`StringBuilder sb = new StringBuilder();`

One way to instantiate the array, is with the new keyword, much as we've seen, with most of the classes we've used to date, with the exception of String.

On this slide, we have an array declaration on the left of the equals sign, and then an array creation expression on the right side.

For comparison, I'm showing you a typical array variable declaration, and a class instance, or object creation expression, using the StringBuilder class.

Instantiating an Array

Array Creation	Object Creation
`int[] integerArray = new int[10];`	`StringBuilder sb = new StringBuilder();`

They look pretty similar, but there are two major differences.

Square brackets are required when using the new keyword, and a size is specified between them. So in this example, there will be 10 elements in the array.

An array instantiation doesn't have a set of parentheses, meaning we can't pass data to a constructor for an array.

In fact, using parentheses with an array instantiation, gives you a compiler error.

Invalid Array Creation – Compile Error because of ()
`int[] integerArray = new int[10]();`

An Array is NOT Resizable.

The size of an array, once created, is fixed.

In this case, integerArray will have 10 elements.

Array Creation

```
int[] integerArray = new int[10];
```

You can't change the size of an array, after the array is instantiated.

We can't add or delete elements, we can only assign values to one of the ten elements in this array, in this example.

The array initializer

An array initializer, makes the job of instantiating and initializing a small array, much easier.

The array initializer

```
int[] firstFivePositives = new int[]{1, 2, 3, 4, 5};
```

In this example, you can see we still use the new keyword, and have int, with the square brackets.

But here we specify the values, we want the array to be initialized to, in a comma delimited list, in curly braces.

Because these values are specified, the length of the array can be determined, so we don't specify the size in the square brackets.

And actually, Java provides an even simpler way to do this.

The array initializer as an anonymous array

Java allows us to drop **new int[]**, from the expression, as we show here.

This is known as an anonymous array.

Here we're showing examples for both an int array, as well as a String array.

The array initializer
`int[] firstFivePositives = {1, 2, 3, 4, 5};`
`String[] names = {"Andy", "Bob", "Charlie", "David", "Eve"};`

An anonymous array initializer, can only be used in a declaration statement.